**چه چیزی پیش روی شما و اسکرام است؟**

شما از پس آن بر آمدید you made it

شما در مورد معایب توسعه plan-driven در فصل های قبل مطالعه کردید، همین طور با متد توسعه آبشاری آشنا شدید. با توسعه plan-driven ما نتوانستیم ارزشی را برای مشتری هایمان ایجاد کنیم یا نتوانستیم در مورد کارهایمان و در مورد آنچه نیاز داریم با مشتری صراحت داشته باشیم. ما نیاز داشتیم تا نرم افزار در حال توسعه را بهبود ببخشیم. ما نمیخواستیم که 400 صفحه مستند طراحی راهی برای حل مشکلات پروژه باشد. ما نیاز داشتیم که مشتریانمان را در خط مقدم تلاشهایمان قرار دهیم.

شما در مورد توسعه value- driven و قهرمان آن agile مطالعه کردید.شما همه چیز را مورد اسکرام آموختید و تعریفی دقیق از xp بدست آوردید. شما مثال های Agile را و اینکه چگونه در دنیای واقعی به کار می آید را مشاهده کردید. آموختید که چگونه ایده های agile را برروی تیم خود اعمال کنید.حتما این سوال برای شما پیش آمده است که ما به کجا میرویم؟ where do we go from here?

بسیار خب ابتدا به جایگاه کنونی مان نگاهی بیاندازیم.

**اسکرام**

اسکرام از نقطه نظر فرهنگی و توسعه نرم افزار برای هدایت تیم توسعه یک چهارچوب سالم است. اسکراممسئول پاسخ گویی و transparency اعضای تیم است. اسکرام مشکلات را خیلی زود تر از پروسه شناسایی میکند اما آنها را حل نمی کند. اسکرام اینکه از چه فریمورک یا سورس کنترلی استفاده کنید یا اینکه چگونه توسعه بدهید را نمی آموزد، همچنین به شما نشان نمی دهد که چگونه کار کنید. اسکراماز صحت ارزش های مشخص شده در agile manifesto همان طور که در ادامه می بینید اطمینان حاصل می کند:

* افراد و تعاملات بین آنها فراتر از پروسه ها و ابزار آلات هستند.
* نرم افزاری که کار میکند فراتر از مستندات جامع است.
* همکاری مشتری فراتر از عقد قرار داد است( مذاکره برای قرار داد)
* پاسخ به تغییر فراتر از اجرای طرح است.

طى مطالعه اين مورد، پروسه ما و تك تك ما، تكامل يافتيم ما با یک انتقاد قابل سازنده آغاز کردیم. چرا ما از همه این نوت های (post it) استفاده میکنیم؟ swim lane ها در روی برد kanban چه چیزی هستند؟ چرا ما هر روز یک جلسه ایجاد میکنیم؟ چرا ما باید داستان های کاربری را در یک فرمت مشخص بنویسیم؟ چرا ما باید زمان زیادی را برای تست های اتوماتیک صرف کنیم؟ من فقط میخواهم سخت بر روی کد خود کار کنم! ما توسعه دهنده هستیم و ما باید موارد خود را به وسیله post-it نوت و بورد مدیریت کنیم؟ به نظر می رسد که ما در زمان یک گام به عقب برداشته ایم!

با این حال ما قدردان post-it نوت ها و swim lane هستیم، چرا که بهترین راه برای به دست گرفتن مدیریت و قابل اندازه گیری کردن قسمت هایی از کار بود و همه افراد تیم را به همراه پیشرفت تیم به روز میکرد. این ابزار به ما اجازه میداد تا چارت های بزرگ و قابل دسترس (BVC) مانند چارت های burn-down بسازیم تا هم اعضای تیم و هم مردمی که در خارج از تیم قرار دارند یک دید کلی از پیشرفت پروژه داشته باشند. ما داستان کاربری را در یک راه مشخص برای کمک به یافتن نیازمندی ها یادداشت میکنیم. و این توافق همچنین برای تبدیل این نیازمندی ها به فیچر های تکرار شونده به ما کمک می کند.

تست اتوماتیک بسیار مهم است چرا که توسعه دهندگان می توانند با کسب و کار و همچنین با تستر ها برای مشخص کردن یک تعریف واضح از نیازمندی ها، ابتدا نوشتن تست ها و بعد کد ها کار کنند. بر عکس اینکه ابتدا کد ها نوشته شود. اگرچه این کار تضمین میکند که کد ها انچه که برای انجام کار میخواهیم هستند، اما شاید با خواسته های کاربر کاملا مطابق نباشد. اینجا جایی است که ارتباط با کاربر وارد عمل می شود. تست اتوماتیک همچنین به تولیدات بزرگ و shippable code به منظور تایید اینکه آیا تست کار میکند و اینکه کد همانی است که کسب و کار میخواهد کمک میکند.

اسکرام یک حامی بزرگ از حلقه بازخورد سریع برای بهبود پروسه ها است. همانند تیم با استفاده از مواردی که مطرح شد اسپرینت ها برای ارزش بخشیدن به به این حلقه بازخورد سریع به وجود آمده اند. حلقه های بازخورد انواع کمی دارند. از نقطه نظر مردم، جلسات روزانه، دموی محصول و عطف کننده هایی وجود دارد.از دید کد ، سرور یکپارچگی دائمی وجود دارد.

جلسات روزانه، سریع ساده و موثر هستند. هر کسی می تواند اینکه دیروز چه کاری انجام داده یا امروز قرار است چه کاری انجام دهد و اینکه چه موانعی که تجربه کرده را اطلاع دهد ، اینها شفافیت کار تیم در مورد پیشرفتشان را بالا می برد. هیچ کس قادر نیست چیزی را از تیم مخفی کند.هیچ کسی نمی تواند از چهارچوب کاری خود خارج شود بدون اینکه کسی بداند که او چه کاری میکند. هر روز همه افراد در جلسات در مورد اینکه چه کارهایی انجام داده اند صحبت میکنند.

در روش های سنتی مدیران پروژه یا رئیس ها باید پیش توسعه دهندگان می آمدند و از آنها میخواستند که وضعیت جدید را در اختیارشان بگذارند. در نهایت این اتفاق بار ها وبارها در روز اتفاق می افتاد. متاسفانه اینکار تمرکز توسعه دهندگان را در هنگام کار از بین میبرد. با جلسات روزانه رئیس مدیر پروژه و هر فردی که میخواهد از وضعیت پروژه با خبر شود در جلسه شرکت میکند. چرا که این جلسات روزانه بسیار سریع هستند و در زمان های ثابت برای ارائه روال کاری روزانه بسیار مناسب هستند.

**دموی محصول**

دموی محصول یک راه مناسب برای بهبود اعتماد مشتری و شفافیت کار است. در طی طرح ریزی اسپرینت، تیم تعدادی از فیچر ها را به یک اسپرینت می سپارند. زمانی که دموی محصول در پایان اسپرینت به دست آمد، تیم آن را به مشتری برای این اسپرینت ارئه می دهد. این کار شفافیت کار را بالا میبرد و اعتماد مشتری را بیشتر میکند و رضایتمندی را برای تیم افزایش می دهد. با مشتری برای اینکه تیم چه کاری انجام داده به گفتگو بنشینید و صحبت های او را برای پروژه گوش کنید.این کاربه نفع خود مشتری است و همچنین توسعه دهندگان نیازهای مشتری را می فهمند و بر روی پروژه اعمال می کنند نه اینکه مشتریان در پایان پروژه چیزی که واقعا می خواهند را بگویند. همان طور که در مطالعه این مورد مشاهده کردید، در پایان هر دو هفته تیم توسعه یک جسله برای نمایش کار تیم برگذار میکند.

ساخت نرم افزار فقط ساخت نرم افزار است. در Agile معیار اصلی پیشرفت، تیم است. این ساخت یک چیزی است که مردم قبلا شبیه به آن را ندیده اند. زمانی که تیم جلسات را در پایان 2هفته برگذار می کند یعنی بخشی از پروژه آماده است و تیم و مشتری می توانند در مورد آن به بحث و گفت و گو بنشینند.

همان طور که در مطالعه این مورد مشاهده کردید، برخی از دموهای محصولات برای دادن ایده هایی از سوی مشتری یا از سوی خود تیم الهام بخش هستند. این نوآوری و الهام بخشی در نتیجه همکاری بین تیم و مشتری است. زمانی که این نوآوری مورد تشویق واقع شود، مشتری و تیم باید این را بدانند که تعریف "انجام" را که پیش از این برایش تصمیم گرفته اند هنوز موثر است. هر ایده جدید قبل از اینکه روی آن کار شود، نیاز دارد که آراسته شود و اولویت بندی هایش مشخص گردد. بهترین ایده امروز شاید در روزهای بعد در پایین ترین اولویت قرار بگیرد.

**عطف کننده ها**

عطف کننده های اسپرینت در مطالعه این مورد می تواند بسیار کار آمد باشد. در پایان هر اسپرینت اعضای تیم در مورد اینکه در این اسپرینت چه چیزی درست یا چه چیزی غلط بوده بحث میکنند و تصمیم میگیرند که چگونه این مسائل را در اسپرینت های بعدی حل کنند. داشتن تیم خود داشتن پروسه ای با یک فریم ورک ایده آل است چرا که هریک از اعضای تیم مانند بخشی از یک چیز هستند و اجباری برای انجام کارهای نا خواسته نیست!

ایجاد یک محیط قابل اعتماد و امن برای اعضای تیم برای بحث و گفت گو بسیار مهم است. ممکن است دقت کرده باشین که برخی افراد خاص تمایل دارند که دایما در مورد موضوعات مختلف بحث کنند در حالی که بقیه به سادگی در پشت صحنه پنهان میشوند(یعنی تمایلی برای بحث کردن ندارند.). برای ایجاد یک محیط قابل اعتماد اسکرام مستر یا اعضای تیم باید زمانی که کسی کنترل جلسه را در دست گرفته اطلاع دهند و آن را برای صحبت دیگران متوقف کنند. کسانی که تمایلی برای بحث کردن ندارند برای تیم بهتر است که آنها را تشویق به صحبت کردن کنند. مدافع آنها برای صحبت کردنشان باشید تا بتوانند چیزی را که فکر می کنند را به زبان بیاورند. شما باید این را بدانید که افراد کم حرف دید عمیق تری به مسائل دارند.

اگر اعضای تیم این احساس را داشته باشند که حتی اگر اعتماد به نفس داشته باشند و صحبت کنند هیچ مشکلی حل نمی شود، عطف کننده ها بی فایده می شوند. ارتباطات خوب، امن و شفاف باعث رشد تیم و رشد هر پروسه ای می شود. برای کمک به آنها نیاز به مهارت های اصلاحی داریم. به طور معمول این مهارت ها را می توان در اسکرام مستر یا agile coach یافت. اما این کار، برای آنها محدود نشده است. هر کدام از اعضای تیم می توانند این مهارت ها را دشته باشند. عطف کننده ها راه خارق العاده ای برای داشتن تیم و بهبود پروسه ها می باشند.

آیتم های کاری که در عطف کننده ها مشخص شده اند، شامل اهمیت برنامه نویسی pair و اهمیت داشتن مشتری های در دسترس یا صاحب محصول. از تغییر گاه به گاه فعالیت های خود در مورد اداره کردن عطف کننده های خود مطمئن شوید. شما میبینید که تیم شروع به انجام عطف کننده می کند و سپس بازی the soup را برای اینکه درک بهتری از موارد داشته باشند انجام می دهند. هرچیزی بعد از چندین بار انجام دادن، کسل کننده می شود و در این صورت مشکلات دیده نخواهند شد مگر اینکه این مشکلات، از زاویه های مختلف بروز دهند. اگر عطف کننده ها کسل کننده شوند، نباید آنها را ادامه داد چرا که فقط باعث اتلاف وقت می شوند. با اینکه توصیه نمی شود، به دلیل عدم مفید بودن عطف کننده ها، آنها را کنارگذاشت. تا زمانی که می توانیم عطف کننده های موثری در پایان هر اسپرینت داشته باشیم حذف آنها نمی تواند برای پروژه سودی داشته باشد؛ چرا که انواع فعالیت ها در طول اسپرینت وجود دارند که برخی از آنها خوب انجام شده اند و برخی دیگر بالعکس.

شما مجبور به تغییر انجام عطف کننده ها در هر اسپرینت نیستید اما تلاش کنید چیز های مختلفی را امتحان کنید. اگر یکی را انجام دادید و جواب نداد میتوانید چیز دیگری را امتحان کنید. اگر شما اسکرام مستر تیم باشید، کتاب ها و وبلاگ هارا چک کنید. برای مثال کتاب Agile Retrospectives از Esther Derby و Diana Larsen یک منبع عالی برای پیدا کردن راه های دیگری برای انجام عطف کننده ها به شمار می رود. این منابع می توانند بازدهی ایده ها را برای اینکه چگونه عطف کننده ها را انجام دهیم افزایش دهد.

همین طور مطمئن شوید که آیتم های کاری که از عطف کننده ها به دست آمده اند، انجام شوند و خیلی سریع حل شوند. اگر آیتم های کاری مشابه بعد از هفته ها دوباره دوباره تکرار شوند این بسیار درد آور است هیچ چیزی بیشتر از اینکه افراد تیم نتیجه کارهایشان را نبینند، به روحیه افراد صدمه نمیزند. باید هر کدام از این آیتم های کاری را به یک نفر از اعضای تیم اختصاص داد، حتی اگر آن کار مربوط به همه اعضای تیم باشد. اطمینان حاصل کردن از اینکه عملیات به درستی انجام شده مسئولیت شخصی است که آن را برعهده گرفته است. اگر فعالیتی زمان بیشتری از یک اسپرینت بگیرد آن را از اسپرینت بیرون می اندازند. بدیهی است که آن برای پروژه زیاد مهم نیست. اما اگر واقعا مورد مهمی باشد دوباره به موارد باز میگردد. با کامل کردن آیتم های کاری، تیم پروژه احساس پیشرفت و اعتماد به نفس می کنند. ارائه راه هایی برای اصلاح پروسه و ندیدن نتیجه آن روحیه اعضای تیم را به طور کامل از بین می برد. بدون کار تیمی باید به راه های قدیمی انجام کارها بازگردیم. به یاد داشته باشید agile برای یافتن ضعف ها عالی است، اما حل آن ضعف ها برعهده شماست.

یکپارچگی مداوم

یک سرور یکپارچه سازی مداوم حلقه بازخورد از دید کد است.این باز خورد برای کل تیم بسیار جدی است بله کد میتواند حرف بزند. کد چیز ها را اینگونه می گوید:

* من نمیتوانم build شوم یک خطا یا یک مورد مرجعی وجود دارد.
* تست های اتوماتیک من شکست خورده اند.
* تست ها مدتی طول می کشد تا کامل شوند.بهتر است سازماندهی این تست ها را در گروه های دیگر و در زمان های دیگر شروع شود کمی بعد از هر ساخته شدن و در زمان های مشخصی از روز.

سرور یکپارچه سازی مداوم توسط توسعه ای با ذخیره سازی زمان توسعه دهندگان به دست می آید.جای استفاده از راه های قدیمی که هر کسی کد را بعد از روز ها یا هفته ها پس از توسعه آن چک میکرد و به وسیله ادغام نا سازگاری هایش با آن سرو کله میزند و هیچ راهی برای فهمیدن شکستن تغییرات بخش های اساسی ندارد، این چطور است که باشکستن کار و اجازه دادن به هر نفر که در یکی از شاخه های کد اصلی کار کند، سرور یکپارچه سازی را در هر بار چک کردن تنظیم کند و در سطوح خیلی کوچکتر با ناسازگاری ها سروکله بزند. همین طور سرور یکپارچه سازی میتواند این را تایید کند که قابلیت های جدید قابلیت های موجود را نقض نمی کند. ارائه سریع این باز خورد اعتماد به نفس را در اعضای تیم القا میکند.

Because the feedback loop is tighter, a lot of the merge conflicts disappear because everyone is working in and checking into the same code base at more frequent intervals.

By running the automated tests after every check in, you get the added benefit of having a stable environment that has already been smoke-tested. The code is ready for the team members that are testing to begin without first having to do regression testing to make sure the new functionality did not break the existing functionality.

**Plan-do-study-act**

بر اساس مطالعه این مورد، ما چرخه deming را دنبال کردیم که با نام پروسه 4 مرحله ای بهبود مداوم ، PSDA یا Plan-do-study-act به رسمیت شناخته می شود.

Plan: در هر اسپرینت یک بخش طرح ریزی برای مشخص شدن اینکه تیم دقیقا میخواهد چه کاری انجام دهد، وجود دارد. داستان های کاربری نوشته می شوند و تخمین زده می شوند و معیار مقبولیت برای تیم و مشتری برای اینکه هر کدام نقش های خود را تا پایان فیچر بدانند نوشته می شود.

Do: این یک اسپرینت است. در طول اسپرینت فیچر ها توسعه می یابند.

Study: در پایان هر اسپرینت، برای یافتن راهی برای بهبود پروسه تیم عطف کننده ای را ایجاد میکند. اعضای تیم از خودشان میپرسند که ما چطور میتوانیم موثرتر واقع شویم و چطور می توانیم پروسه خود را بهبود ببخشیم؟ به عنوان نتیجه این مطالعه، فعالیت ها یادداشت می شوند تا بعدا اجرا شوند.

Act: اینجا فعالیت هایی که در بخش عطف کننده ها مشخص شده اند، وجود دارند. تیم تاثیر بزرگی بر روی پروسه دارد. فعالیت ها از بخش عطف کننده انتخاب می شوند و تیم آنها را برای بهبود پروسه، بر روی آن اعمال میکند.

این پروسه برای هر اسپرینت تکرار می شود. تیم ثابت کرده است که می تواند ....

**Xp**

زمانی که اسکرام مدیریت پروژه است و پروسه بخشی از پزوژه است، xp یک پروسه یا یک متدولوژی برای توسعه فیچر ها و ابزاری برای انجام کار به شمار می آید. بیایید نگاهیی به 5 ارزش xp بیاندازیم ودر مورد آنها بحث کنیم.

جلسات روزانه همچنین این شکاف ارتباطی را بین اعضای تیم متصل می سازند. باتسهیل این جریان ارتباطی بین تیم و خارج از آن آنها توانستند موارد و موانع را مشخص کنند،آنها را بیرون از راه بیاندازند و به ادامه مسیر توسعه نرم افزار و اضافه کردن ارزش برای نرم افزار مشتری می پردازد.

*Communication*: The success or failure of a team largely depends on communication. As

was determined early in the case study, Sarah became a bottleneck in the process due to

her domain knowledge. This was communicated by Tyler since he spoke up and informed

the group he couldn't finish a story due to waiting on Sarah. This proved to be constructive

criticism as they needed to share domain knowledge. The spreading of domain knowledge

across the team was accomplished through pair programming.

* سادگی: از آنجایی که سارا و تایلر هر دو و حتی جو، توسعه دهندگان نرم افزار خوبی هستند، آنها قادر هستند با هم ارتباط برقرار کنند و طوفان مغزی ایجاد کنند؛ در نهایت نیتجه یک راه حل ساده می شود. راه حل های خوب معمولا نیتجه تفکر اولیه نیستند. به وسیله برنامه نویسی pair، برنامه نویس ها میتوانند با یکدیگر مشورت کنند برای آنکه یک راه حل ساده تر بیابند. سادگی محصول را قابل نگه داری تر برای آینده می سازد. شما می دانید که کد زمانی نوشته می شود که به آن نیاز است. تیم کارشان را با یک فریم ورک معماری (mvc,NHibernate,وغیره...) که بر اساس دانش تیم و نیازمندی های سطح بالا تصمیم میگیرند، آغاز میکند. فیچر های اولیه و کم برای اثبات و ایجاد اعتماد به نفس سپری می شود. با گذشت زمان معماری از تیم ایجاد شد که زمانی که نیاز داشتند از آن استفاده میکردند، نه یک طرحی برای چیزی که حتی اتفاق نیفتاده. Agile manifesto میگوید سادگی- هنر به حداکثر رساندن مقدار کاری که انجام می دهید نیست- ضروری است.
* Feedback: Feedback came from the time and effort the developers spent writing tests so as

not to break anything. The feedback loop from a code perspective using a CI server can

save countless hours by providing you quick feedback as opposed to trying to fix a build a week after five developers checked different things in over the course of time.

شجاعت : تیم باید برای برقراری ارتباط با مشتریانی که در مورد فیچر ها ایده هایی خارج از محدوده دارند، شجاعت داشته باشد.به جای بی سروصدا به عقب بازگشتن و توسعه این فیچر ها تیم بایدشجاعت برقراری ارتباط داشته باشد که ایده های جدید خارق العاده هستند، اما نیاز داریم که بر روی آنها کار کنیم چرا که آنها جزیی از داستان نیستند. این کار اهمیت تعریفی برای انجام کار(پایان کار)، قبل از نوشته شدن فیچر ها را اثبات می کند، چرا که هر کسی انتظاراتی دارد.

Respect: While respect can come in many different ways, one example comes from team

members breaking the build. Tyler had an issue with that early on; and while the team did

not yell at him, they did bring it to his attention. Tyler began to pay closer attention, broke

fewer builds, and became more respectful of the team. This openness and willingness to

accept constructive criticism lead to a respectful team that could focus on work and not

drama.